Методичні вказівки
до виконання практичних робіт з дисципліни
«Розробка проєктів засобами платформи .NET».
Частина 1. «Фреймворк машинного навчання ML.NET»
зі спеціальності «Інженерія програмного забезпечення»

Міністерство освіти і науки України
Вінницький національний технічний університет

Методичні вказівки до виконання практичних робіт з дисципліни «Розробка проєктів засобами платформи.NET». Частина 1. «Фреймворк машинного навчання ML.NET» зі спеціальності «Інженерія програмного забезпечення»

Вінниця
ВНТУ
2026

Рекомендовано до видання Радою з якості освіти Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 6 від 18.12.2025 р.)

Рецензенти:

Л. В. Крупельницький, кандидат технічних наук, доцент, доцент кафедри обчислювальної техніки ВНТУ

В. О. Денисюк, кандидат технічних наук, доцент, доцент кафедри комп'ютерних наук ВНТУ

Методичні вказівки до виконання практичних робіт з дисципліни «Розробка проєктів засобами платформи .NET». Частина 1. «Фреймворк машинного навчання ML.NET» зі спеціальності «Інженерія програмного забезпечення» / уклад. Г. Б. Ракитянська. Електрон. текст. дані. Вінниця : ВНТУ, 2026. 36 с.

У методичних вказівках наведено основні теоретичні дані до виконання практичних робіт з дисципліни «Розробка проєктів засобами платформи.NET» засобами технологій ML.NET. Розглядаються задачі прогнозування часових рядів, побудови рекомендаційних систем, класифікації зображень, класифікації тексту та аналізу зображень сцен. Такий підхід дозволяє опанувати ключові етапи роботи з ML.NET: формування навчального набору даних, створення та налаштування конвеєра обробки текстової інформації, тренування моделі та класифікацію нових описів. Практична спрямованість проєктів сприяє розвитку навичок створення інтелектуальних сервісів та формує розуміння принципів роботи сучасних моделей машинного навчання.

ЗМІСТ

Вступ

Стрімкий розвиток цифрових технологій та широке впровадження інтелектуальних інформаційних систем обумовлюють зростання ролі методів машинного навчання у сучасній програмній інженерії [1-3]. Системи класифікації зображень, аналізу сцен, розпізнавання об’єктів та автоматичної інтерпретації медіаконтенту дедалі частіше використовуються у мобільних застосунках, сервісах безпеки, транспортних системах, цифрових медіаплатформах та в інших сферах, де необхідна швидка та надійна обробка складних даних за допомогою моделей глибинного навчання [4-7]. Це вимагає від фахівців глибоких знань не лише у сфері розробки програмного забезпечення, але й у прикладних методах Data Science [8, 9]. Особливо актуальним є застосування компактних моделей машинного навчання для роботи на ресурсно обмежених пристроях, таких як мобільні телефони та вбудовані системи. Такі моделі будують із застосуванням технологій глибинного та трансферного навчання [10-12].

Платформа ML.NET є ключовим інструментарієм, що дозволяє легко інтегрувати засоби машинного навчання у .NET-застосунки [13]. Вона дає змогу розробникам створювати та навчати моделі, виконувати попередню обробку даних, будувати конвеєри обчислень і прогнозувати результати без необхідності звернення до зовнішніх сервісів. Завдяки простоті використання та гнучкості, ML.NET є ефективною базою для навчання студентів основам побудови інтелектуальних компонентів програмних систем.

У даних методичних вказівках розглядається задача побудови моделі класифікації сцен на основі текстових описів. Такий підхід дозволяє опанувати ключові етапи роботи з ML.NET: формування навчального набору даних, створення та налаштування конвеєра обробки текстової інформації, тренування моделі та класифікацію нових описів. Практична спрямованість проєкту сприяє розвитку навичок створення інтелектуальних сервісів та формує розуміння принципів роботи сучасних моделей машинного навчання.

Методика навчання моделі аналізу зображень сцен ґрунтується на результатах наукових досліджень, виконаних аспірантом Прусом Богданом Вікторовичем у дисертаційній роботі на тему «Моделювання та оптимізація надійності програмної системи для аналізу зображень сцен на пристроях з обмеженими обчислювальними ресурсами». Запропонований метод будується на основі дистиляції знань із глибинної мережі в систему нечітких гранулярних прототип-орієнтованих правил [14, 15]. У роботі розроблено підходи до лінгвістичного опису сцен, прототип-орієнтованого подання ознак та оптимізації моделей для роботи на ресурсно обмежених пристроях. Інтеграція окремих результатів дисертації у навчальний матеріал забезпечує актуальність методики, поглиблює прикладний зміст практичної роботи та сприяє формуванню зв’язку між освітнім процесом і сучасними науковими досягненнями у галузі програмної інженерії.

ТЕМА № 1. Розробка системи прогнозування засобами ML.NET

Мета: здобути навички роботи із сценаріями ML.NET «Аналіз часових рядів (Time Series Analysis)» та «Прогнозування величин (Value Prediction)»; розробити модель бази даних; розробити програмну модель прогнозування часового ряду; навчити та оцінити точність моделі.

Загальні відомості

Для ефективного ведення діяльності, управління запасами відіграє ключову роль. Занадто велика кількість товару на складі означає, що непродані товари зберігаються, але не приносять жодного доходу. Занадто мала кількість потенційних клієнтів призводить до втрати продажів та купівлі товарів у конкурентів. Тому питання полягає в тому, яка оптимальна кількість запасів, яку слід тримати в наявності? Аналіз часових рядів допомагає знайти відповідь на ці питання, розглядаючи історичні дані, виявляючи закономірності та використовуючи цю інформацію для прогнозування значень у майбутньому. Одновимірний аналіз часових рядів розглядає одне числове спостереження протягом певного періоду часу з певними інтервалами, такими як погодинний або щомісячний попит.

Методом аналізу даних є одновимірний аналіз часових рядів за допомогою алгоритму сингулярного спектрального аналізу (SSA) [13]. SSA передбачає розкладання часового ряду на набір головних компонентів. Ці компоненти можна інтерпретувати як складові сигналу, які відповідають трендам, паттернам (подіям), сезонності та багатьом іншим факторам. Компоненти підлягають реконструкції, утворюючи часове вікно, та використовуються для прогнозування значень на деякий час у майбутньому (горизонт прогнозування). Аналогічну задачу вирішує регресійна модель: прогнозування на основі факторів впливу. Також дана задача дотична до задачі виявлення подій (патернів) та аномалій в даних, де прогнозний показник може бути пов’язаний з іншими подіями. Ці дані також використовують для перевірки правильності алгоритмів прогнозування часового ряду.

В ML.NET прогнозний конвеєр ForecastingPipeline приймає 365 точок даних за перший рік та вибірково розподіляє набір даних часового ряду на 30-денні (щомісячні) інтервали, як зазначено параметром SeriesLength. Для визначення прогнозованого значення на наступний період, використовуються значення за попередні сім днів (вікно прогнозування). Модель налаштована на прогнозування семи наступних діб, які визначено параметром горизонту. Оскільки прогноз є обґрунтованим припущенням, тому необхідно знати діапазон значень у найкращому та найгіршому сценаріях, визначений верхньою та нижньою межами. У цьому випадку рівень довіри для нижньої та верхньої меж встановлено на рівні 95%. Рівень довіри можна відповідно збільшити або зменшити. Чим вище значення, тим ширший діапазон між верхньою та нижньою межами для досягнення бажаного рівня достовірності.

Структура набору даних

Прикладом часового ряду є Bike Sharing Dataset [16]. Використовуючи системи велопрокату, користувач може орендувати велосипед на певній точці та повернути його на іншій. Сьогодні існує великий інтерес до цих систем через їхню важливу роль у питаннях дорожнього руху, навколишнього середовища та здоров'я. Характеристики даних, що генеруються системами спільного користування велосипедами, роблять їх важливими для досліджень. На відміну від інших транспортних послуг, таких як автобус чи метро, у цих системах чітко фіксується тривалість поїздки, місце відправлення та місце прибуття. Ця функція перетворює систему спільного користування велосипедами на віртуальну сенсорну мережу, яку використовують для визначення мобільності в місті.

Процес оренди велосипедів тісно пов'язаний з екологічними та сезонними умовами [16]. Погодні умови, опади, день тижня, пора року, година доби тощо можуть впливати на поведінку орендарів. Дані агрегуються щогодини та щодня, а потім додається інформація про погоду та сезонність.

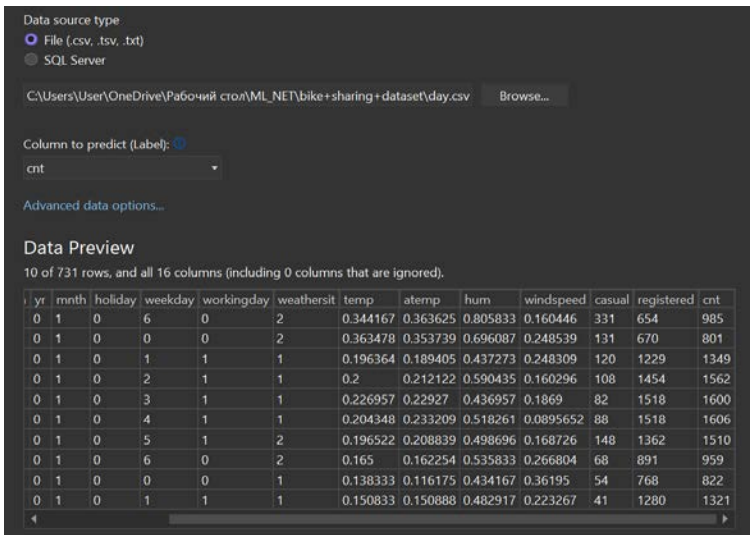
Вхідними параметрами моделі прогнозування є: x_1 – дата; x_2 – сезон (1: весна, 2: літо, 3: осінь, 4: зима); x_3 – місяць (1 - 12); x_4 – година (0 - 23); x_5 – святковий період (1) або ні (0); x_5 – день тижня; x_6 – робочий день (1) або вихідний (0); x_7 – погодні умови (1: ясно, мінлива хмарність, 2: туман, хмарно з проясненнями, 3: невеликий дощ та сніг , хмарно, 4: сильний дощ, мокрий сніг, туман; x_8 – температура повітря; x_8 – вологість повітря; x_9 – швидкість вітру; x_{10} – кількість випадкових користувачів; x_{11} – кількість зареєстрованих користувачів. Прогнозним показником є y – загальна кількість орендованих велосипедів.

Набір даних нараховує дані оренди велосипедів в місті за два роки спостережень (731 дні), 17379 рядків для погодинної оренди [16].

Розробка моделі прогнозування в ML.NET включає етапи:

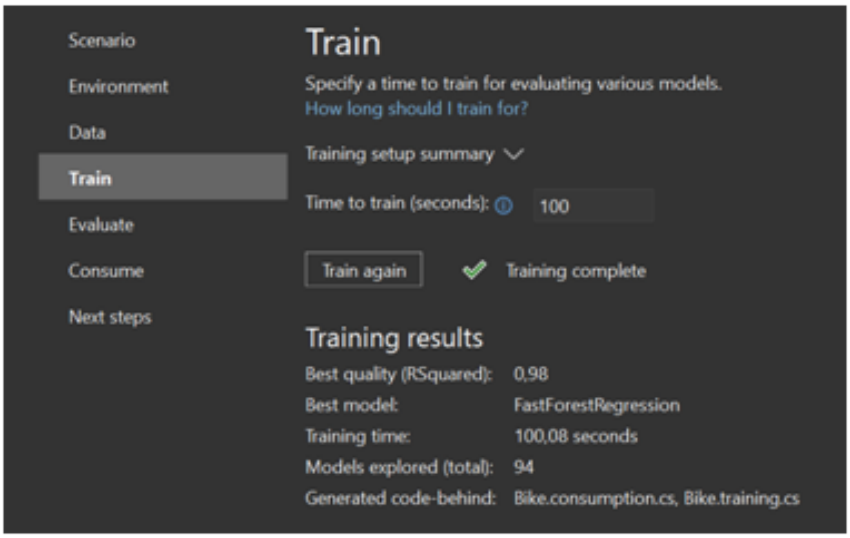
1. Обрати сценарій Model Builder «Value Prediction». Додати MachineLearning Model до проєкту. Сценарій підтримується Local CPU.

2. Задати структуру даних – ознаки (features) і прогнозний показник.

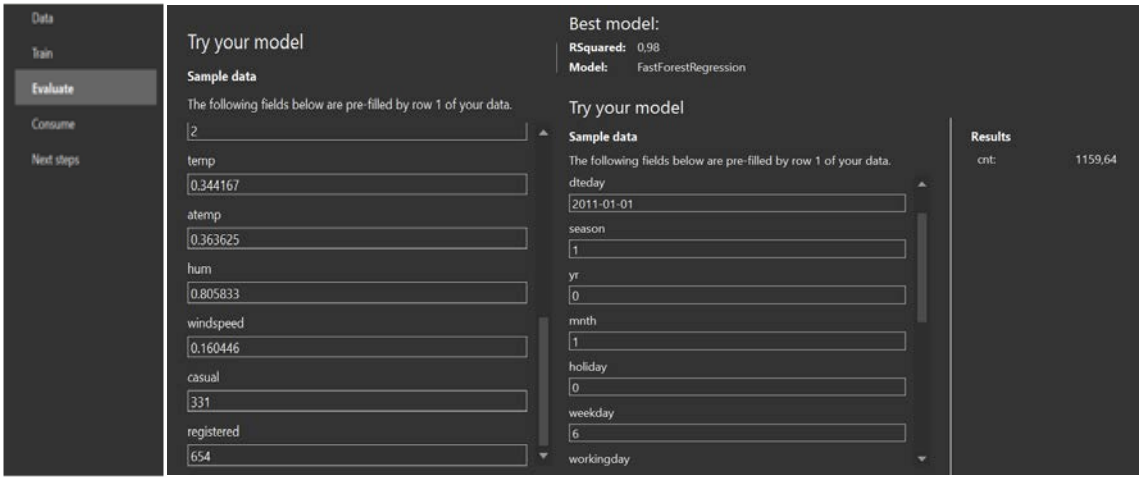


	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
0	1	0	4	1	1	0.204348	0.233209	0.518261	0.0895652	88	1518	1606
0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362	1510
0	1	0	6	0	2	0.165	0.162254	0.535833	0.266804	68	891	959
0	1	0	0	0	1	0.138333	0.116175	0.434167	0.36195	54	768	822
0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280	1321

3. Навчити модель (Train). Обрати модель, що забезпечує максимальну точність на даному наборі даних (мінімальну різницю між прогнозованою та реальною кількістю замовлень у кожний проміжок часу). Оцінити середньо-квадратичну похибку (RSquered) та час навчання.



4. Прогнозування за допомогою моделі (Evaluate). Для точки вибірки $\mathbf{X} = (x_1 – x_{12})$ значення cnt = 1160.



5. Додати до проєкту (Add to Solution) згенерований код для навчання моделі (Train) та прогнозування (Consumption).

– модель навчання конвеєра RetrainPipeLine (конфігурація процесу обробки даних з перетвореннями даних конвеєра ITransformer)

```
public partial class Bike
{
    0 references
    public static ITransformer RetrainPipeline(MLContext context, IDataView trainData)
    {
        var pipeline = BuildPipeline(context);
        var model = pipeline.Fit(trainData);
        return model;
    }
    1 reference
    public static IEstimator<ITransformer> BuildPipeline(MLContext mlContext)
    {
        var pipeline = mlContext.Transforms.ReplaceMissingValues(new []
            {new InputOutputColumnPair(@"instant", @"instant"),
             new InputOutputColumnPair(@"season", @"season"),
             new InputOutputColumnPair(@"yr", @"yr"),
             new InputOutputColumnPair(@"mnth", @"mnth"),
             new InputOutputColumnPair(@"casual", @"casual"),
             new InputOutputColumnPair(@"registered", @"registered")})
             .Append(mlContext.Transforms.Text.FeaturizeText(@"dteday", @"dteday"))
             .Append(mlContext.Transforms.Concatenate(@"Features", new []{@"instant",@"season",
             @"yr",@"mnth",@"holiday",@"weekday",@"workingday",@"weathersit",@"temp",@"atemp",
             @"hum",@"windspeed",@"casual",@"registered",@"dteday"}))
             .Append(mlContext.Regression.Trainers.FastForest(new FastForestRegressionTrainer.Options
             {NumberOfTrees=158,FeatureFraction=0.1F,LabelColumnName=@"cnt",
                 FeatureColumnName=@"Features"}));
        return pipeline;
    }
}
```

– модель прогнозування PredictEngine

```
public partial class Bike
{
    model input class

    model output class

    private static string MLNetModelPath = Path.GetFullPath("Bike.zip");
    public static readonly Lazy<PredictionEngine<ModelInput, ModelOutput>>
        PredictEngine = new Lazy<PredictionEngine<ModelInput,
            ModelOutput>>(() => CreatePredictEngine(), true);
    1 reference
    public static ModelOutput Predict(ModelInput input)
    {
        var predEngine = PredictEngine.Value;
        return predEngine.Predict(input);
    }
    1 reference
    private static PredictionEngine<ModelInput, ModelOutput> CreatePredictEngine()
    {
        var mlContext = new MLContext();
        ITransformer mlModel = mlContext.Model.Load(MLNetModelPath, out var _);
        return mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);
    }
}
```

Завдання

Розробити C# .NET Core консольний додаток для прогнозування попиту на велосипеди за допомогою сценарію аналізу одновимірних часових рядів ML.NET на основі даних, що зберігаються в базі даних SQL Server.

1. Розробити модель бази даних. Вихідний набір даних зберігається у таблицях за такою схемою в базі даних SQL Server:

```
CREATE TABLE [Rentals]
([RentalDate],[WeatherConditions],[Registred],[UnregularDemand],[TotalRentals])
```

Завантажити дані для навчання з бази даних.

2. Створити модель прогнозування. Визначити конвеєр аналізу часових рядів `forecastingPipeline`:

```
var forecastingPipeline = mlContext.Forecasting.Bike_Rent(
outputColumnName: "Forecasted_Rent",
inputColumnName: "Total_Rent",
windowSize: 7,
seriesLength: 30,
trainSize: 365,
horizon: 7,
confidenceLevel: 0.95f,
confidenceLowerBoundColumn: "Lower_Bound_Rent",
confidenceUpperBoundColumn: "Upper_Bound_Rent");
```

3. Отримати інтервальні значення прогнозного показника

```
Date: 1/10/2025
Actual Rentals: 2530
Lower Estimate: 1293.64
Forecast: 2285.74
Upper Estimate: 3541.18
```

Для різних значень confidenceLevel отримати границі значень прогнозного показника. Оцінити точність прогнозування за допомогою RSquered.

4. Здобути знання із даних часового ряду у вигляді причинно-наслідкових лінгвістичних висловлювань «патерни (події) – прогноз», де патерни описують тенденції даних часового вікна, а прогноз – патерни горизонта.

Контрольні питання

1. Які моделі аналізу даних використовують для обробки часових рядів?
2. Які етапи включає побудова моделі прогнозування в ML.NET?
3. Що визначають вікно та горизонт прогнозування в методі SSA?
4. Які патерни визначають тренди часового ряду? Які фактори впливають на події у часових рядах?
5. Які методи DataPreprocessing використовує ML.NET?
6. Які показники визначають точність прогнозу?
7. Як отримати інтервальні оцінки значення прогнозного показника? Як встановлюється рівень довіри?
8. Як залежить інтервал прогнозних значень від рівня довіри?
9. Які багатовимірні задачі аналізу даних можуть бути зведені до задачі прогнозування величин?
10. Як здобути оптимальну підмножину ознак для прогнозування?

ТЕМА № 2. Розробка рекомендаційної системи засобами ML.NET

Мета: набути навички роботи із сценарієм ML.NET «Рекомендаційні системи (Recommendations)»; розробити програмну модель на основі методу матричної факторизації: задати структуру набору даних «користувачі-пропозиції», розробити модулі для навчання та прогнозування моделі, оцінити точність прогнозування рейтингу для системи рекомендацій.

Загальні відомості:

Існує декілька методів вирішення задачі проєктування рекомендаційних систем, таких як рекомендація списку фільмів, музики або інших пропозицій та продуктів [13]. В цьому випадку необхідно спрогнозувати, яку оцінку в діапазоні [1, 5] користувач надасть певному продукту, і рекомендувати цей продукт, якщо рейтингова оцінка перевищує визначений поріг (чим вищий рейтинг, тим вища ймовірність того, що користувачеві сподобається певний продукт). Основна ідея методу матричної факторизації полягає у знаходженні двох факторних матриць низького рангу для апроксимації навчальної матриці. Навчальні дані (факторизована матриця) – це список кортежів. Кожен кортеж складається з індексу стовпця, індексу рядка та значення рейтингу, визначеного цими двома індексами. Всі записи в навчальній матриці не можуть бути вказані, оскільки матриця є розрідженою. Для заповнення пропущених значень використовують матричну факторизацію

Припустимо, що ідентифікатори користувачів User_ID та ідентифікатори продуктів Product_ID використовуються як індекси рядків та стовпців відповідно, а значення матриці – це оцінки, надані користувачами відповідним продуктам. Тобто, оцінка r у рядку u та стовпці v означає, що користувач u надає рейтинг r елементу v. Матриця оцінок є неповною, оскільки користувачі не можуть надавати свої відгуки про всі відомі продукти, які можуть налічувати сотні тисяч найменувань.

Нехай $\boldsymbol{R} \in \mathbb{R}_{m \times n}$ – матриця рейтингів, де m – кількість зареєстрованих користувачів, n – кількість рекомендованих продуктів. Матриця рейтингів $\boldsymbol{R}$ може бути подана у вигляді композиції двох факторних матриць $\boldsymbol{P} \in \mathbb{R}_{k \times m}$ та $\boldsymbol{Q} \in \mathbb{R}_{k \times n}$, де k – ранг апроксимації. Прогнозований рейтинг у рядку u та стовпці v матриці $\boldsymbol{R}$ буде скалярним добутком рядка u матриці $\boldsymbol{P}$ та рядка v матриці $\boldsymbol{Q}$, тобто $\boldsymbol{R}$ апроксимується добутком транспонованої матриці $\boldsymbol{P}^{\mathrm{T}}$ та $\boldsymbol{Q}$. Ранг апроксимації k обирається набагато меншим за m та n, тому добуток $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{Q}$ називають низькоранговою апроксимацією $\boldsymbol{R}$. Метод навчання будується на основі стохастичного градієнтного методу та методу координатного спуску для знаходження $\boldsymbol{P}$ та $\boldsymbol{Q}$ шляхом мінімізації відстані між матрицею $\boldsymbol{R}$ та її наближенням $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{Q}$. Метод координатного спуску призначений для однокласової матричної факторизації, де всі спостережувані рейтинги є цілими числами або значення рейтингу дорівнюють 0 або 1.

Для багатопотокової реалізації стохастичного градієнтного методу використовують паралельний стохастичний градієнтний метод для матричної факторизації в системах зі спільною пам'яттю [13]. Спосіб декомпозиції матриці оцінок визначає розклад швидкості навчання всередині потоку. Метод паралельного координатного спуску та формула декомпозиції для однокласової факторизації матриці будується на основі методу відбору негативних зразків факторної матриці. ML.NET використовує бібліотеку для паралельної факторизації матриці в системах зі спільною пам’яттю.

Прикладом є набір даних recommendation-ratings-train.csv [17]. Матрична факторизація є алгоритмом для навчання рекомендаціям, коли є дані про те, як користувачі оцінювали продукти в минулому. Ознака TimeStamp може бути видалена як надлишкова. У цьому випадку алгоритм матричної факторизації використовує метод колаборативної фільтрації. Якщо User_1 та User_2 оцінюють переглянуті фільми Item 1,…, Item n однаково, то User_2, з високою ймовірністю, сподобається фільм Item n+1, який User_1 переглянув і високо оцінив:

Movie	Item 1	Item n	Item n+1
User 1	$r = 5$	$r = 5$	$r = 4$
User 2	$r = 4$	$r = 5$	Has not watched, RECOMMEND movie

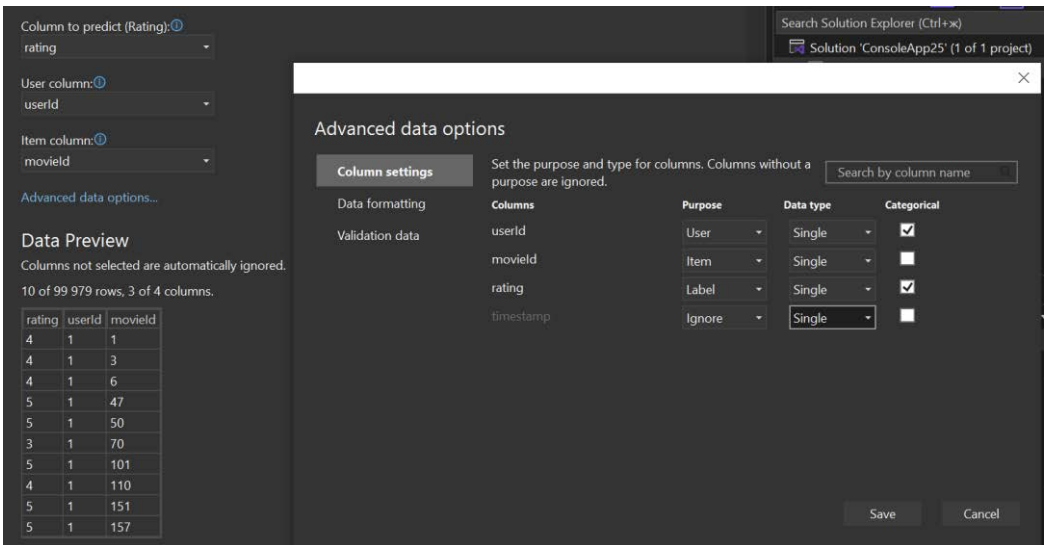
Поширеною проблемою у колаборативній фільтрації є проблема сценарію нового користувача, коли відсутні попередні дані про належність користувача до певних кластерів [13]. Ця проблема вирішується шляхом створення профілю або історії оцінювання користувача з рейтингами контенту, який користувач переглядав раніше. На основі контекстних історій реалізується алгоритм виведення прогнозних рейтингів для нових продуктів.

ML.NET використовує факторизацію матриці одного класу (One Class Matrix Factorization), коли відомі лише user_Id та movie_Id. Цей метод рекомендацій базується на сценарії ймовірного спільного замовлення на основі аналізу історій замовлень користувачів. ML.NET використовує машини факторизації з урахуванням полів (Field Aware Factorization Machines) для надання рекомендацій, коли окрім user_Id, movie_Id та рейтингу є більше ознак (наприклад, текстовий опис та ціна продукту). Цей метод використовує колаборативний підхід фільтрації.

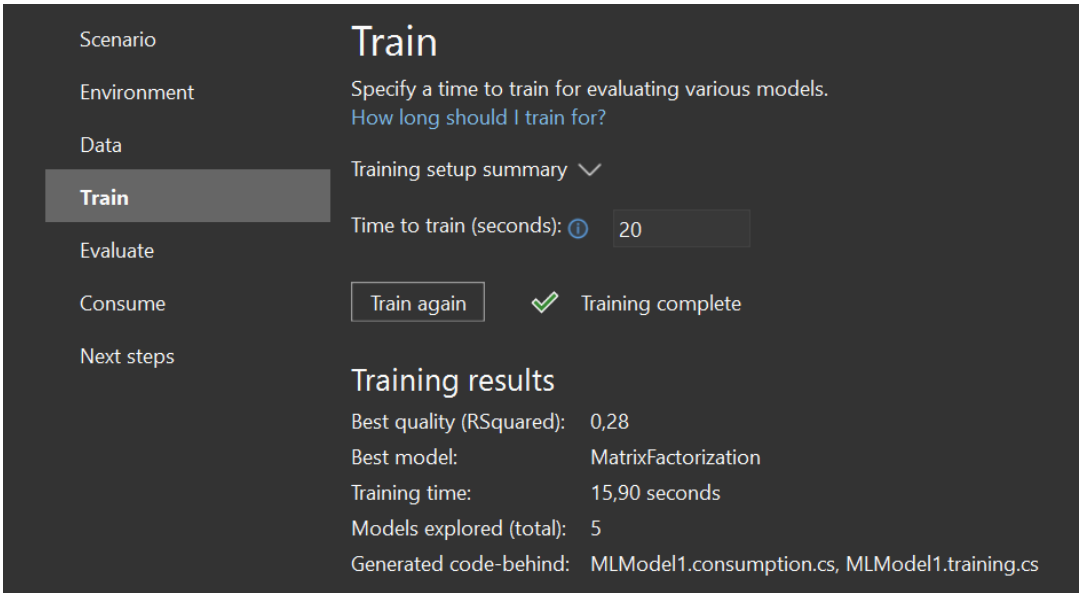
Розробка моделі рекомендацій в ML.NET включає етапи:

1. Обрати сценарій Model Builder «Recommendation». Додати MachineLearning Model до проєкту. Сценарій підтримується Local CPU.

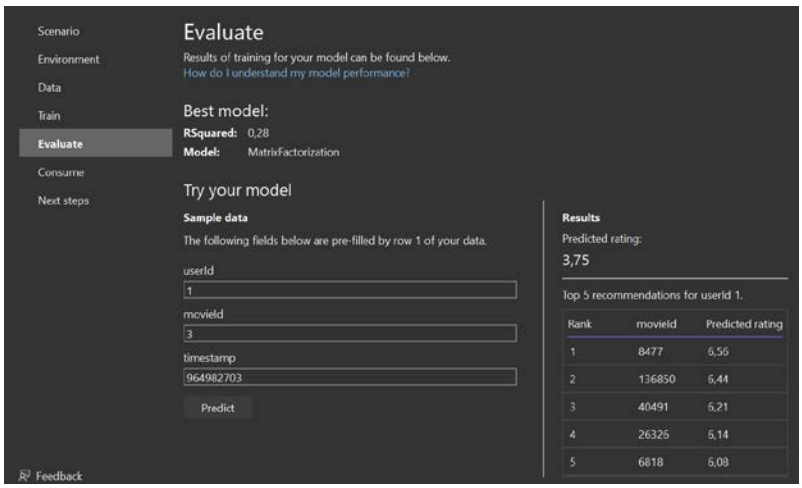
2. Задати структуру набору даних (user_Id, movie_Id, опис Item) і обрати прогнозний показник Rating. Завантажити набір даних.



3. Навчити модель (Train). Задати MatrixFactorization(Options) для методу однокласової матричної факторизації. Обрати модель, що забезпечує максимальну точність на даному наборі даних (мінімальну різницю між прогнозованим та фактичним рейтингом). Оцінити середньоквадратичну похибку апроксимації матриці оцінок (RSquered) та час навчання.



4. Прогнозування рейтингу за допомогою моделі (Evaluate). Для точки вибірки [User_Id = 1, Movie_Id = 3, Rating = 4] спрогнозувати рейтинг і надати top5 телепрограм для перегляду. За прогнозом моделі рейтинг склав Rating = 3.75. Значення у вибірці Rating = 4, що свідчить про достатню для практичного застосування точність навчання. Рейтинг top5 телепрограм, рекомендованих для перегляду UserId = 1 складає від 6.08 до 6.56.



5. Додати до проєкту (Add to Solution) згенерований код для навчання моделі (Train.cs) та прогнозування (Consumption.cs):

– модель навчання конвеєра RetrainPipeLine (конфігурація процесу обробки даних з перетвореннями даних конвеєра ITransformer)

```
public partial class MLModel_Recommend
{
    0 references
    public static ITransformer RetrainPipeline(MLContext context, IDataView trainData)
    {
        var pipeline = BuildPipeline(context);
        var model = pipeline.Fit(trainData);
        return model;
    }
    1 reference
    public static IEstimator<ITransformer> BuildPipeline(MLContext mlContext)
    {
        var pipeline = mlContext.Transforms.Conversion.MapValueToKey(@"userId", @"userId")
            .Append(mlContext.Transforms.Conversion.MapValueToKey(@"movieId", @"movieId"))
            .Append(mlContext.Recommendation().Trainers.MatrixFactorization
            (labelColumnName:@"rating",
            matrixColumnIndexColumnName:@"userId",
            matrixRowIndexColumnName:@"movieId"));
        return pipeline;
    }
}
```

– модель прогнозування PredictEngine, яка завантажує модель для прогнозування на основі одного кортежа даних:

```
public static ModelOutput Predict(ModelInput input)
{
    var predEngine = PredictEngine.Value;
    return predEngine.Predict(input);
}

1 reference
private static PredictionEngine<ModelInput, ModelOutput> CreatePredictEngine()
{
    var mlContext = new MLContext();
    ITransformer mlModel = mlContext.Model.Load(MLNetModelPath, out var _);
    return mlContext.Model.CreatePredictionEngine<ModelInput, ModelOutput>(mlModel);
}
```

Завдання

1. Розробити сценарій нового користувача. Створити профіль користувача для побудови історії замовлень на основі попередніх оцінок.

Розробити систему рекомендацій із розширеним набором ознак: x_0 – UserID; x_1 – вік користувача; x_2 – уподобання жанрів; x_3 – пристрій для перегляду; x_4 – вихідний (робочий) день; x_5 – діапазон часу перегляду.

2. Розробити метод кластеризації користувачів. Порівняти методи пониження вимірності матриці «користувачі - пропозиції».

3. Оцінити точність прогнозування рейтингу контенту для набору даних movie.txt. Дослідити залежність похибки від набору ознак.

4. До ознак додати «Текстовий опис (анотація)». Згенерувати словник і навчити модель прогнозувати рейтинг за текстовим описом. Вибірку даних сформувати на основі текстових анотацій та imdb-рейтингу контенту.

5. Розробити метод ранжування пропозицій для формуванняTop5 List за множиною критеріїв. Задачу багатокритеріального оцінювання формалізувати у вигляді задачі класифікації «критерії - рейтинг (1, 2, …, 5)».

Контрольні питання

1. Які методи використовують для побудови рекомендаційних систем?
2. Який мінімальний набір ознак для системи рекомендацій?
3. Який цільовий показник прогнозує модель рекомендацій?
4. Які ознаки має матриця «користувачі - пропозиції»?
5. Які існують методи пониження вимірності матриці ознак?
6. Як побудувати рекомендаційну систему засобами ML.NET?
7. Коли використовують метод головних компонент?
8. Який набір ознак для методу колаборативної фільтрації?
9. Як надається обґрунтування згенерованого recommendation list?
10. Який показник оцінює точність програмної моделі?

ТЕМА № 3. Класифікація тексту на основі технологій ML.NET

Мета: набути навички роботи із сценарієм ML.NET «Text Classification»; включити модель TensorFlow у конвеєр ML.NET; навчити та оцінити модель ML.NET; класифікувати текстові дані.

Загальні відомості

Метод ML.NET навчання моделі класифікації тексту на основі бінарної логістичної регресії IEstimator<TTransformer> використовує стохастичний метод двоїстого координатного сходження. Навчена модель калібрована та може генерувати ймовірність, передаючи вихідне значення лінійної функції. Навчання моделі базується на методі стохастичного двоїстого координатного сходження (SDCA), сучасному методі оптимізації опуклих цільових функцій. Алгоритм може бути масштабований, оскільки це алгоритм потокового навчання. Збіжність методу забезпечується шляхом періодичної синхронізації між первинними та двоїстими змінними в окре-

мому потоці. Використовується кілька варіантів функцій втрат, таких як втрати ознак та логістичні втрати. Залежно від використаної функції втрат, модель може навчатись методом опорних векторів або логістичною регресією. Метод SDCA поєднує декілька переваг, таких як можливість потокового навчання (без завантаження всього набору даних у пам'ять), досягнення допустимого розв’язку шляхом кількох проходжень усього набору даних та відсутність обчислень з нульовим результатом на розріджених наборах даних.

SDCA є стохастичним та потоковим алгоритмом оптимізації [13]. Результат залежить від порядку навчальних даних, оскільки критерій зупинки не є жорстким. У строго опуклій оптимізації оптимальний розв’язок єдиний, тому всі запуски досягають однієї допустимої області. Навіть коли поверхня цільової функції є мультимодальною, метод забезпечує допустимі розв’язки при повторних запусках. В цьому випадку для повторних результатів параметри методу пошуку становлять Shuffle = False, NumThreads = 1. Клас SdcaLogisticRegressionBinaryTrainer використовує емпіричну мінімізацію ризиків (ERM) для формулювання задачі оптимізації на основі наявних даних. Емпіричний ризик вимірюється шляхом застосування функції втрат до прогнозів моделі на основі точок вибірки. Якщо навчальні дані не містять достатньої кількості точок (для навчання моделі в n-вимірному просторі потрібна вибірка об’ємом R^n), може статися перенавчання, в результаті чого модель, створена ERM, добре описуватиме навчальні дані, але втратить здатність до узагальнення і не зможе передбачити правильних результатів у невідомих областях.

Регуляризація є методом згладжування такого явища шляхом штрафування вимірюваної функції норми параметрів моделі [13]. Метод навчання ML.NET підтримує регуляризацію еластичної мережі, яка штрафує лінійну комбінацію регуляризацій L_1-норми (LASSO) та L_2-норми для вектора параметрів **W**. Регуляризація L_1-норми та L_2-норми мають різний ефект та способи використання, які доповнюють один одного. Зокрема, алгоритм оптимізації із регуляризацією L_1-норми може збільшити розрідженість вагових коефіцієнтів моделі **W**. Для багатовимірних та розріджених наборів даних, ретельно підібрані коефіцієнти L_1-норми дозволяють досягти високу точність прогнозування з моделлю, яка має малу частку ненульових ваг (0.01 від загальної кількості ваг моделі). На відміну, L_2-норма не збільшує розрідженості навченої моделі, але може запобігти перенавчанню, уникаючи великих значень параметрів. В окремих випадках, коли використання L_2-норми призводить до кращої якості прогнозування, налаштуванню підлягають коефіцієнти L_1-норми та L_2-норми. Концептуально використання L_1-норми означає, що розподіл усіх параметрів моделі є розподілом Лапласа, тоді як L_2-норма передбачає для них Гауссівський розподіл. Надмірна регуляризація (присвоєння великих коефіцієнтів регуляризації L_1-норми або L_2-норми) може погіршити точність прогнозування, виключаючи важливі змінні з моделі. Тому вибір правильних коефіцієнтів

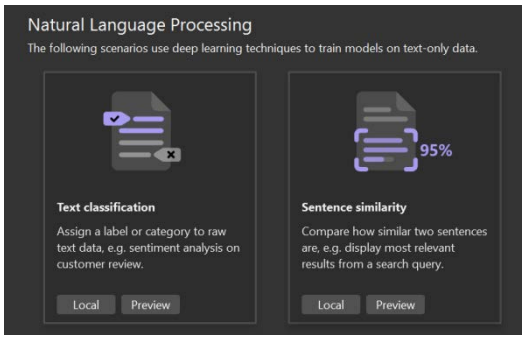
регуляризації є важливим на практиці для запобігання перенавчання моделі.

Прикладом є задача визначення тональності тексту у коментарях користувачів [18]. Клас вхідного набору даних SentimentData має рядок для коментарів користувача (SentimentText) та булеве значення (Class) 1 (позитивний) або 0 (негативний) для відгуку. Модель здобуває ознаки із тексту і класифікує текстові дані за допомогою моделі логістичної регресії.

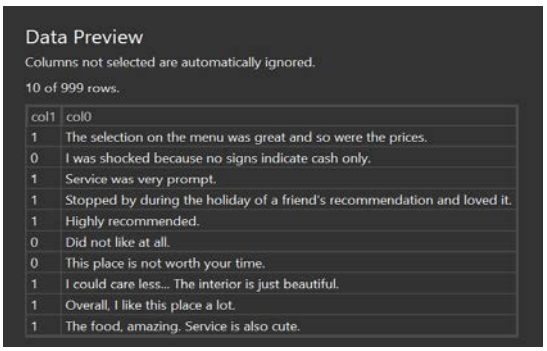
У задачах класифікації тексту, ML.NET використовує метод перестановки ознак (Permutation Feature Importance, PFI) для визначення важливості ознак моделі [13]. PFI працює із позначеним набором даних, вибираючи ознаку та переставляючи значення для цієї ознаки в усіх прикладах, так що кожен приклад тепер має випадкове значення для ознаки та початкові значення для всіх інших ознак. Потім для зміненого набору даних обчислюється зміна метрики оцінки (RSquered) порівняно з вихідним набором даних. Чим більша зміна в метриці оцінки, тим важливішою є ця ознака для моделі. Обчислення PFI потребує часових витрат, оскільки метрика оцінки розраховується багаторазово для визначення важливості кожної ознаки.

Розробка моделі класифікації тексту в ML.NET включає етапи:

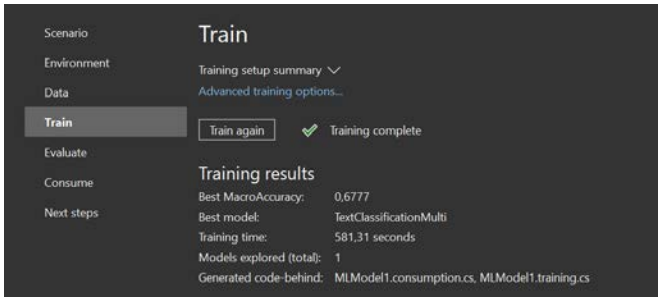
1. Обрати сценарій Model Builder «Text Classification». Додати MachineLearning Model до проєкту. Сценарій підтримується Local CPU.



2. Задати структуру даних – текст (features) і клас (тональність, 1(0))

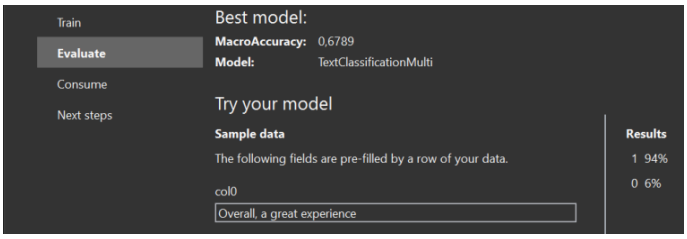


3. Навчити модель (Train). Обрати модель, що забезпечує максимальну точність на даному наборі даних (мінімальну різницю між модельним та експериментальним ступенем належності до класу). Оцінити середньоквадратичну похибку (RSquered) та час навчання.

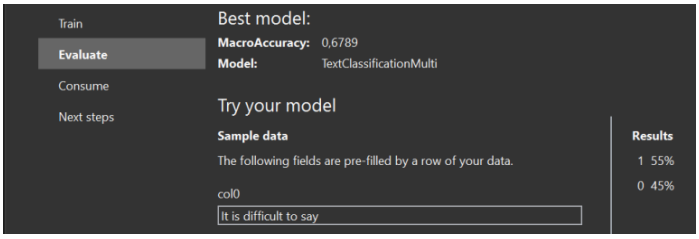


5. Класифікація за допомогою моделі (Evaluate):

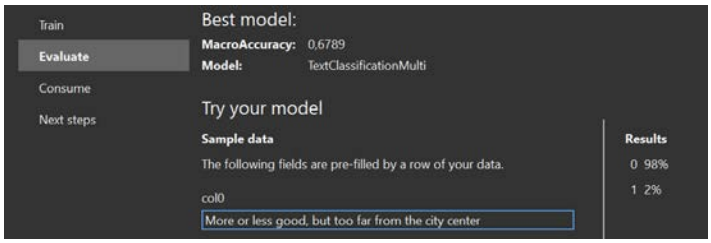
– із строгою належністю до класу (strict membership)



– із частковою належністю до класу (partial membership)



– із строгою неналежністю до класу (non-membership)



Приклади з недостатньою належністю до класу (позитивний або негативний відгук) не можуть бути чітко класифіковані. Такі приклади формують невизначену граничну область між класами, яка може бути класифікована як «нейтральний відгук». Для визначення граничних точок необхідно задати порогові значення для ступеня належності даних до класу [13].

5. Додати до проєкту (Add to Solution) згенерований код для навчання моделі (Train) та прогнозування (Consumption).

– метод здобування ознак (permutation feature importance) – повертає список ознак та їх важливість

```
public static List<Tuple<string, double>> CalculatePFI(MLContext mlContext, IDataView trainData,
    ITransformer model, string labelColumnName)
{
    var preprocessedTrainData = model.Transform(trainData);
    var permutationFeatureImportance = mlContext.MulticlassClassification
.PermutationFeatureImportance(model, preprocessedTrainData, labelColumnName: labelColumnName);

    var featureImportanceMetrics = permutationFeatureImportance
         .Select((kvp) => new { kvp.Key, kvp.Value.MacroAccuracy })
         .OrderByDescending(myFeatures => Math.Abs(myFeatures.MacroAccuracy.Mean));

    var featurePFI = new List<Tuple<string, double>>();
    foreach (var feature in featureImportanceMetrics)
    {
        var pfiValue = Math.Abs(feature.MacroAccuracy.Mean);
        featurePFI.Add(new Tuple<string, double>(feature.Key, pfiValue));
    }
    return featurePFI;
}
```

– модель навчання конвеєра RetrainPipeLine (конфігурація процесу обробки даних з перетвореннями даних конвеєра ITransformer)

```
public static IEstimator<ITransformer> BuildPipeline(MLContext mlContext)
{
    var pipeline = mlContext.Transforms.ReplaceMissingValues(@"col0", @"col0")
        .Append(mlContext.Transforms.Concatenate(@"Features", new[] { @"col0" }))
        .Append(mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: @"col1",
                inputColumnName: @"col1", addKeyValueAnnotationsAsText: false))
        .Append(mlContext.MulticlassClassification.Trainers.OneVersusAll(binaryEstimator:
        mlContext.BinaryClassification.Trainers.FastTree(new FastTreeBinaryTrainer.Options()
            { NumberOfLeaves = 4, MinimumExampleCountPerLeaf = 20, NumberOfTrees = 4,
              MaximumBinCountPerFeature = 254, FeatureFraction = 1, LearningRate = 0.01,
              LabelColumnName = @"col1", FeatureColumnName = @"Features",
              DiskTranspose = false }), labelColumnName: @"col1"))
       .Append(mlContext.Transforms.Conversion.MapKeyToValue(outputColumnName:
      @"PredictedLabel", inputColumnName: @"PredictedLabel"));
    return pipeline;
}
```

Завдання

Розробити застосунок ASP.NET, який класифікує відгуки користувачів на веб-сайті в режимі реального часу. При побудові конвеєра навчання використовувати попередньо навчену модель TensorFlow [19-21].

1. Моделювання даних. Перетворити текст коментарів вебсайту у вектор ознак для навчання моделі.

Відгуки користувачів – це текст у вільній формі. Програма перетворює текст у вхідний формат моделі. Необхідно розділити текст на окремі слова та встановити відповідність кожного слова з цілочисельним кодуванням слів у словнику. Результатом цього перетворення є цілочисельний масив змінної довжини, що відповідає кількості слів у реченні. Масив ознак змінної довжини змінюється до фіксованої довжини 600 слів:

PROPERTY	VALUE	TYPE
ReviewText	«this film is really very good»	string
VariableLengthFeatures	[640, 43, 56, 537, 1020, 682]	int[600]

Словник для кодування слів здійснюється за допомогою методу LoadFromTextFile, що дозволяє перетворити текст у вектор ознак.

2. Завантажте попередньо навчену модель TensorFlow та Microsoft.ML NuGet packages: Microsoft.ML.TensorFlow, Microsoft.ML.SampleUtils, SciSharp.TensorFlow.Redist.

```
TensorFlowModel tensorFlowModel = mlContext.Model.LoadTensorFlowModel(_modelPath);
```

3. Використайте модель для класифікації тексту. Вихід моделі TensorFlow формується у вигляді Softmax – ступеня належності до класу в інтервалі [0, 1]. Порогові значення задаються параметром Threshold = 0.5. Метод PredictionEngine дозволяє класифікувати один зразок даних і може використовуватись в однопотокових або прототипних середовищах.

4. Додайте код для створення моделі з PipeLine. Модель ML.NET створюється з ланцюжка оцінювачів у конвеєрі шляхом виклику методу Fit.

Контрольні питання

1. Які моделі аналізу текстових даних використовує сценарій ML.NET?
2. Які етапи включає побудова моделі класифікації тексту в ML.NET?
3. Які методи здобування ознак використовуються в ML.NET для класифікації тексту? У чому суть методу перестановок?
4. В чому суть методу навчання на основі логістичної регресії?
5. В чому суть методу стохастичного двоїстого координатного сходження (SDCA)? Як забезпечується збіжність методу?
6. Як уникнути перенавчання моделі за допомогою регуляризації?
7. Які особливості методів L_1 та L_2-регуляризації?
8. Які етапи навчання за допомогою моделі TensorFlow? Які існують методи побудови словника для дескриптора ознак?
9. Які показники визначають точність класифікації? Як інтерпретуються дані з недостатньою належністю до класів?
10. Як навчаються порогові значення для методу логістичної регресії?

ТЕМА № 4. Класифікація зображень засобами ML.NET

Мета: набути навички роботи із сценарієм ML.NET «Image Classification»; опанувати технологію трансферного навчання; включити попередньо навчену модель TensorFlow у конвеєр ML.NET; навчити та оцінити модель ML.NET; класифікувати тестові зображення.

Загальні відомості

Класифікація зображень – це задача комп’ютерного зору, яка приймає зображення як вхідні дані та класифікує його за заданим класом. Моделі класифікації зображень навчаються за допомогою глибокого навчання

нейронних мереж, зокрема ML.NET використовує технологію трансферного навчання [13]. Щоб навчити власну глибинну модель, використовують попередньо навчену модель TensorFlow і ML.NET Image Classification API. Трансферне навчання застосовує знання, отримані в результаті вирішення однієї проблеми, до вирішення іншої пов'язаної проблеми.

Навчання моделі глибинного навчання з нуля вимагає встановлення мільйонів параметрів, безлічі маркованих навчальних даних та значної кількості обчислювальних ресурсів (сотні годин роботи на GPU). Використання попередньо навченої моделі дозволяє скоротити цей процес, працюючи з тисячами зображень замість мільйонів позначених зображень, і досить швидко створювати власну модель (протягом години без GPU). Передача знань від наперед навченої моделі здійснюється шляхом побудови архітектури Teacher-Student на основі відповіді мережі (response-based), ознак (feature-based) та екземплярів (response-based). Важливо не тільки як розмічені дані (GroundTrueLabels), але як класифікує точки вибірки масштабна попередньо навчена модель-вчитель. Тоді компактна користувацька модель-учень вчиться імітувати відповіді моделі та формувати вектори ознак.

В процесі дистиляції знань від масштабної попередньо навченої моделі (Teacher Model) у компактну модель (Student Model) можлива втрата точності через стиснення моделі (quantization and pruning) шляхом пропорційного зменшення мережі або відсікання окремих блоків. Порівняно із початковим набором зображень, трансферне навчання масштабує процес, використовуючи лише частку навчальних зображень (target domain). Втрати при дистиляції оцінюють метрикою DistillationLoss.

Модель TensorFlow Inception навчена класифікувати зображення за тисячею категорій. Здатність моделі Inception розпізнавати та класифікувати зображення використовується при побудові класифікатора із обмеженими категоріями. Оскільки модель ML.NET навчена розпізнавати шаблони на зображеннях, вона може використовувати частину цих шаблонів у своєму конвеєрі для здобування ознак з нових зображень. Попередньо навчена модель – це 101-шарова версія Residual Network (ResNet) v2. Модель приймає на вхід зображення розміром 224 x 224 та виводить ймовірності для кожного з класів, на яких вона навчалась. Частина цієї моделі використовується для навчання нової моделі на користувацьких зображеннях і класифікує зображення для меншого набору категорій.

API класифікації зображень розпочинає процес навчання, завантажуючи попередньо навчену модель TensorFlow. Процес навчання складається з двох етапів: фаза Bottleneck і фаза навчання. Під час фази Bottleneck завантажується набір навчальних зображень, а значення пікселів використовуються як вхідні дані або ознаки для заморожених шарів попередньо навченої моделі. Заморожені шари включають усі шари нейронної мережі до передостаннього шару, який називають шар Bottleneck. На заморожених шарах не відбувається навчання, а операції виконуються наскрізним способом. Саме на заморожених шарах обчислюються шаблони нижчого рів-

ня, які допомагають моделі розрізняти різні класи. Чим більша кількість шарів, тим більше обчислювальних ресурсів потребує цей крок. Оскільки це одноразовий розрахунок, результати можна кешувати та використовувати в наступних запусках під час експериментів з різними параметрами.

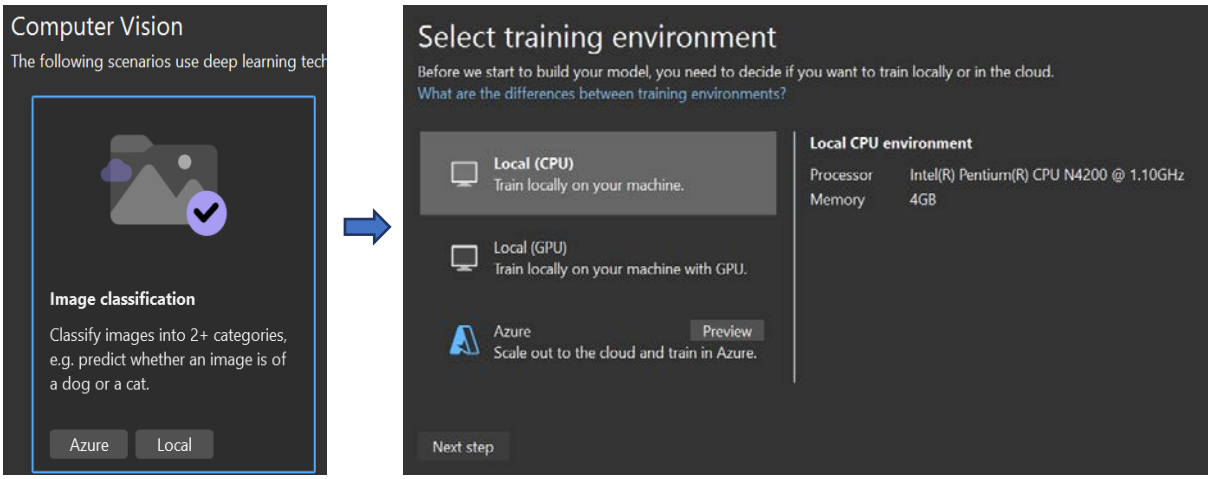
Розраховані вихідні значення з фази Bottleneck використовуються як вхідні дані для перенавчання останнього шару моделі. Це ітеративний процес, де кількість запусків визначається параметрами моделі. Під час кожного запуску оцінюються втрати дистиляції та точність. Параметри моделі налаштовуються з метою мінімізації втрат та максимізації точності.

Приклад. Для прогнозування обрано набір зображень Flowers DataSet [22]. Загальна кількість зображень 4317; відношення training/testing 80/20.

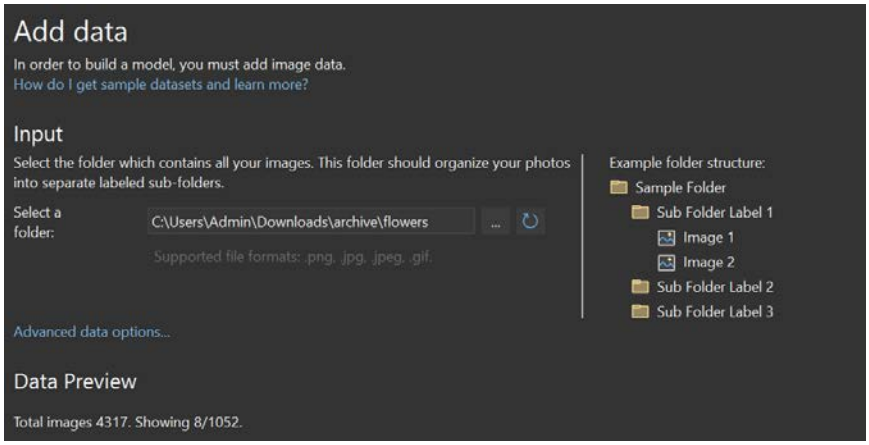
Навчання моделі з використанням сценаріїв ML.NET включає етапи:

1. Створити у VisualSudio консольний застосунок та додати файл моделі машинного навчання. Обрати сценарій ML.NET «Image classification». Даний сценарій використовує технології глибинного (трансферного) навчання для класифікації зображень на дві та більше категорій.

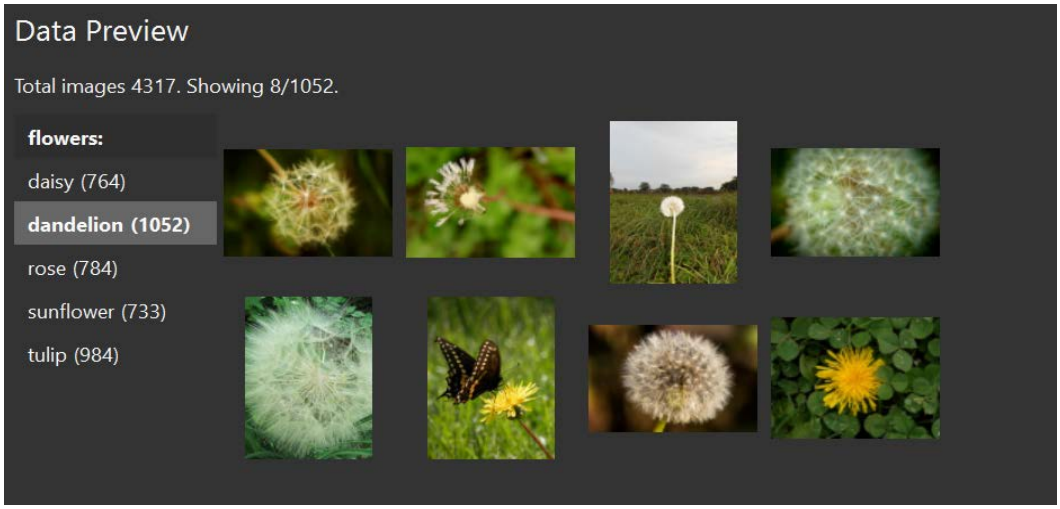
Обрати середовище для навчання (Local CPU, GPU, Azure)



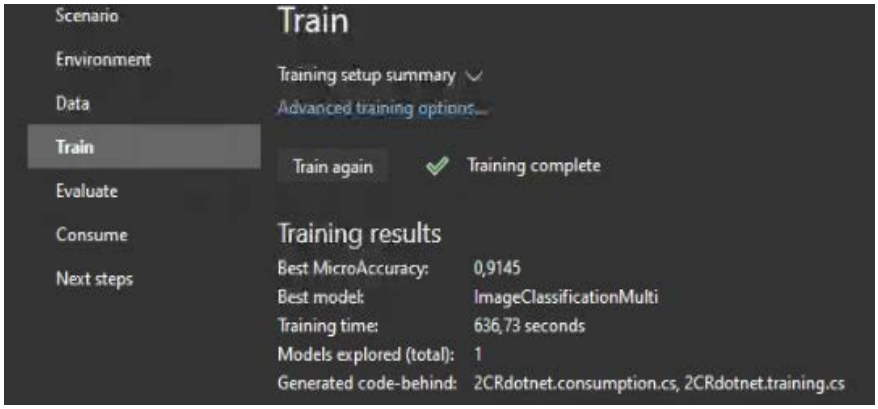
2. Обрати налаштування мережі та завантажити датасет. Задати структуру набору даних та розбити набір зображень по категоріях.



Зображення кожного класу містяться в окремій теці з назвою «Клас»



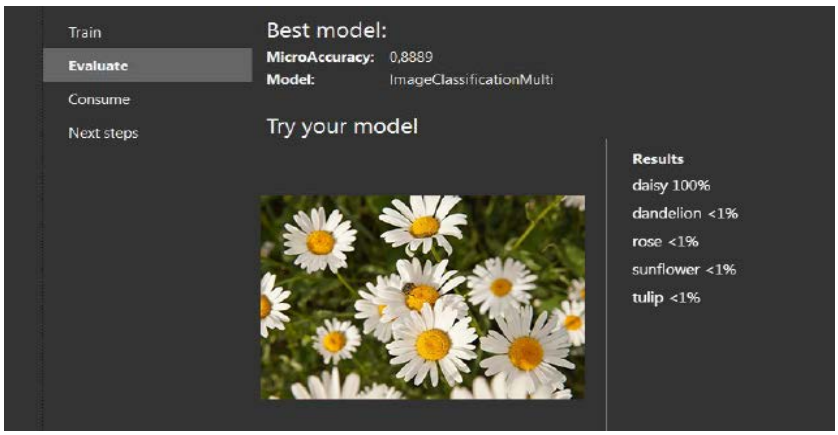
3. Навчити модель. Обрати модель, яка забезпечує найменшу похибку для даного набору даних. Оцінити час навчання.



Після використання моделі TensorFlow для вилучення ознак, придатних як вхідні дані для класичного алгоритму машинного навчання, додаємо багатокласовий класифікатор ML.NET (Add to Solution). Найбільш ефективним навчальним алгоритмом є алгоритм поліноміальної логістичної регресії. Алгоритм добре працює на задачах з великою кількістю ознак, що характерно для моделі глибокого навчання для обробки зображень.

Після завершення навчання генеруються два формати моделі. Одна з них - це версія моделі .pb, а інша - серіалізована версія моделі ML.NET .zip. Під час роботи в середовищах, що підтримуються ML.NET, використовують .zip-версію моделі. У середовищах, де ML.NET не підтримується, використовують версію.pb.

4. Оцінити точність моделі для тестового набору зображень. Критерій RSquered розраховується як відстань між позначками класів (GroundTrue Values) та soft_max вихідного шару трансферної моделі. Метрика DistillationLoss розраховується як відстань між soft_max вихідного шару попередньо навченої моделі TensorFlow і трансферної моделі.



5. Додати до проєкту (Add to Solution) згенерований код для навчання моделі (Train.cs) та класифікації (Consumption.cs):

– модель навчання конвеєра RetrainPipeLine з перетвореннями даних конвеєра ITransformer:

```
/// Retrain model using the pipeline generated as part of the training process.
/// <param name="mlContext"></param>
/// <param name="trainData"></param>
/// <returns></returns>
0 references
public static ITransformer RetrainModel(MLContext mlContext, IDataView trainData)
{
    var pipeline = BuildPipeline(mlContext);
    var model = pipeline.Fit(trainData);

    return model;
}
```

– конфігурація процесу завантаження та перегляду зображень:

```
public static IDataView LoadImageFromFolder(MLContext mlContext, string folder)
{
    var res = new List<ModelInput>();
    var allowedImageExtensions = new[] { ".png", ".jpg", ".jpeg", ".gif" };
    DirectoryInfo rootDirectoryInfo = new DirectoryInfo(folder);
    DirectoryInfo[] subDirectories = rootDirectoryInfo.GetDirectories();
    if (subDirectories.Length == 0)
    {
        throw new Exception("fail to find subdirectories");
    }
    foreach (DirectoryInfo directory in subDirectories)
    {
        var imageList = directory.EnumerateFiles().Where(f =>
        allowedImageExtensions.Contains(f.Extension.ToLower()));
        if (imageList.Count() > 0)
        {
            res.AddRange(imageList.Select(i => new ModelInput
            {
                Label = directory.Name,
                ImageSource = File.ReadAllBytes(i.FullName),
            }));
        }
    }
    return mlContext.Data.LoadFromEnumerable(res);
}
```

Завдання

Розробити консольний застосунок на C# .NET Core, який класифікує зображення за допомогою попередньо навченої моделі глибокого навчання TensorFlow Inseption.

1. Розробити програмну модель на основі трансферного навчання. Включити попередньо навчену модель TensorFlow у конвеєр ML.NET. Завантажити файл параметрів assets .pb.

2. Задати структуру набору зображень. Розробити конвеєр навчання при додаванні нового класу. Розробити алгоритм навчання для частково поміченого набору зображень з послідовним додавання прикладів.

3. Навчити та оцінити модель ML.NET. Розробити модулі для навчання (training.cs) та використання (consumption.cs) моделі.

4. Класифікувати тестові зображення. Оцінити точність класифікації зображень по тестувальній вибірці. Оцінити втрати при дистиляції.

5. Оцінити витрати часу та обчислювальних ресурсів при навчанні з нуля та шляхом трансферного навчання. Оцінити можливість розгортання моделі ML на пристроях з обмеженими обчислювальними ресурсами.

Контрольні питання

1. Які переваги надає технологія трансферного навчання?
2. Які характеристики наперед навченої глибинної нейронної моделі (Teacher Model)? Яку архітектуру має Teacher Model в ML.NET?
3. Які етапи навчання Student Model в ML.NET?
4. Які існують методи стиснення глибинних моделей (методи дистиляції знань)?
5. Які види архітектур (response-based, feature-base, relation-based) використовують для трансферного навчання?
6. В чому суть методу квантизації моделі?
7. Що характеризують знання в архітектурі трансферного навчання?
8. Як мінімізувати втрати точності при дистиляції знань?
9. Які вимоги до об’єму набору зображень для трансферного навчання?
10. Для яких задач комп’ютерного зору використовують технології трансферного навчання?

ТЕМА № 5. Аналіз зображень сцен засобами ML.NET

Мета: набути навички роботи із сценаріями аналізу сцен на основі описів, сформованих за результатами нечіткого структурного представлення зображення; опанувати метод прототип-орієнтованої класифікації сцен; сформувати множину лінгвістичних описів сцен та навчити модель ML.NET класифікувати сцени за цими описами; оцінити точність класифікації та дослідити відповідність прогнозів сцен структурним прототипам.

Загальні відомості:

Аналіз зображень сцен є однією з ключових задач комп’ютерного зору, у якій важливими є не лише окремі об'єкти, а й взаємозв’язки між

ними, їхнє розташування, функціональна взаємодія та композиція сцени. Традиційні підходи до класифікації сцен ґрунтуються на безпосередній обробці піксельних даних за допомогою глибинних нейронних мереж. Проте на пристроях з обмеженими обчислювальними ресурсами повномасштабні моделі глибинного навчання є ресурсно затратними, що вимагає застосування методів компактного подання сцени із вбудовою знань [14, 15].

Одним із сучасних підходів є прототип-орієнтована класифікація сцен, у якій зображення подається у вигляді множини гранул, що описують локальні об’єкти, та гранулярних структур, які характеризують відношення між ними. На основі цих структур формується узагальнений лінгвістичний опис сцени, що визначає її семантичний зміст. Для побудови таких описів використовується система нечітких реляційних рівнянь, яка дозволяє додавати локальні патерни до глобальної структури сцени та відбирати найбільш релевантні прототипи [14, 15].

Лінгвістичний опис – це природномовне твердження, що описує сцену як сукупність об’єктів та просторових відношень між ними (наприклад: *“people with luggage near an airplane”*, *“a person at a table with food and tableware”*). Лінгвістичне представлення є компактним та інтерпретованим, що легко переноситься між пристроями, а також може бути використане як вхідні дані для класичних алгоритмів машинного навчання.

У ML.NET класифікація сцен реалізується шляхом навчання моделі багатокласової класифікації на основі текстових описів сцен. ML.NET автоматично перетворює текст у вектор ознак, будує багатовимірне представлення описів та навчає класифікатор прогнозувати клас сцени [13]. Цей підхід дозволяє поєднувати алгоритми нечіткого логічного виведення, що працюють на рівні об’єктів, із традиційними методами класифікації тексту, забезпечуючи адаптивність системи до нових типів сцен. При цьому властивості об’єктів сцени розглядаються як об’єктно-орієнтовані ознаки.

Структура набору даних:

Для аналізу сцен використовується набір лінгвістичних описів, отриманих у процесі реконструкції сцени. Кожний запис містить:

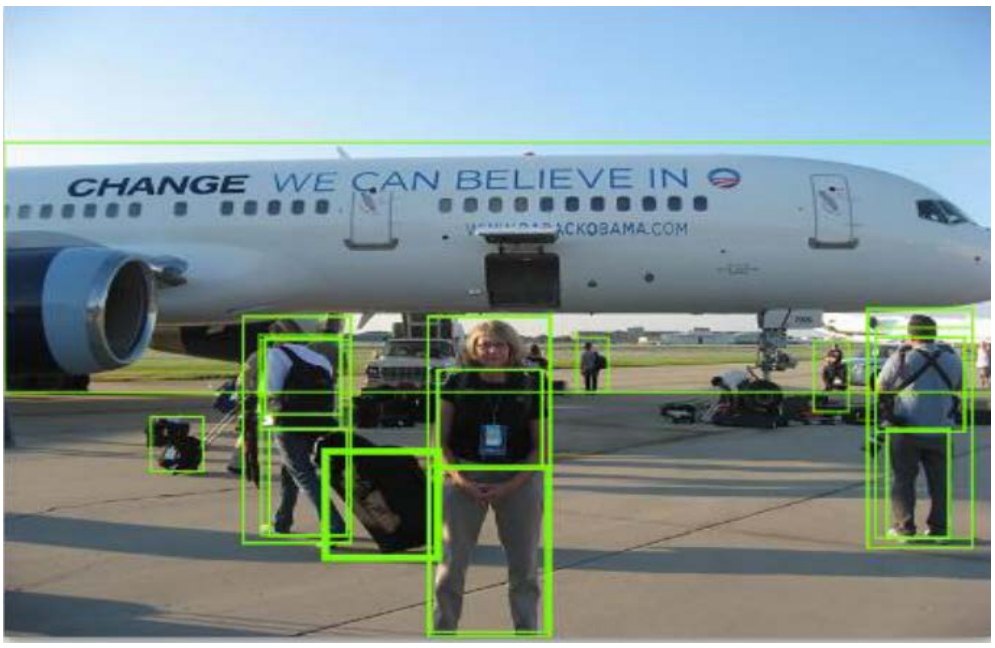
X – текстовий опис сцени, сформований у вигляді прототип-орієнтованих гранулярних структур;

y – категорію сцени (наприклад: Food, Travels, Animals and pets, Portraits, Business, Art тощо).

Опис сцени формується на підставі: кількості та типів об’єктів; відношень між об’єктами (просторових, функціональних, семантичних); відповідності структури сцени одному або кільком прототипам [15]. Відношення «object – subject» формуються на основі статистично усталених сполучень, пов’язаних із діяльністю людини. Наприклад:

SceneText: People with luggage near an airplane

ClassLabel: Travel



Розробка моделі аналізу сцен в ML.NET включає етапи:

1. Формування множини описів сцен.

Створюється вибірка лінгвістичних описів для кожної категорії сцен. Опис відповідає структурі прототипів, визначених у методі прототип-орієнтованої класифікації. Датасет повинен охоплювати різні варіанти одних і тих самих сцен, що забезпечує варіативність мовних конструкцій «object - subject» та підвищує якість моделі.

2. Завантаження набору даних та визначення структури ознак.

У якості вхідного параметра використовується текст сцени, який перетворюється у вектор ознак методом текстової обробки ML.NET (токенізація, нормалізація, *n*-грамні представлення). Категорія сцени визначається як вихідна змінна.

3. Навчання моделі класифікації.

В ML.NET використовується алгоритм багатокласової класифікації, налаштований на роботу із текстовими описами. Метою навчання є мінімізація відстані між прогнозованим та фактичним soft-max виходом моделі для класу сцени. Модель обирається з урахуванням точності (MicroAccuracy, MacroAccuracy) та часу навчання.

4. Класифікація сцен за текстовим описом.

Для нової сцени, отриманої в результаті структурної реконструкції візуальної фрази (granular description), формується текстовий опис. Модель на основі ML.NET визначає найімовірніший клас сцени на основі візуальних відношень «object – subject», які встановлено в результаті навчання.

5. Додавання до проєкту вихідного коду для навчання моделі та прогнозування результатів, аналіз структури конвеєра та параметрів моделі.

Автоматично формується конвеєр обробки тексту та модель прогнозування категорії сцени. Згенерований код додається до проєкту.

Завдання:

1. Сформувати набір лінгвістичних описів сцен, використовуючи структури прототипів «object – subject – relation».
2. Виконати реконструкцію сцени на основі візуальних відношень включення «об'єкти – гранулярні структури – прототипи». Сформувати подання сцени (візуальну фразу) у вигляді матриць відношень включення.
3. Розробити модель багатокласової класифікації сцен засобами ML.NET на основі сформованих описів.
4. Навчити модель та оцінити її точність за показниками MicroAccuracy, MacroAccuracy та LogLoss.
5. Провести класифікацію тестових описів сцен, визначити категорію та порівняти результати з реальними класами.
6. Дослідити залежність точності класифікації від розширення словника описів, кількості варіантів прототипів та різноманітності мовних конструкцій «object – subject – relation».
7. Проаналізувати випадки помилкової класифікації через внутрішньокласові розбіжності (IntraClassBias) та міжкласову схожість (InterClassSimilarity). Визначити, які елементи опису вплинули на похибку.

Приклад

У даному прикладі буде продемонстровано створення моделі класифікації сцен на основі текстових описів із використанням ML.NET. Метою є побудова моделі, яка навчиться розрізняти категорії сцен, описаних словами, та виконувати прогнозування для нового введеного опису.

У середовищі Visual Studio створюємо новий проєкт типу Console App (.NET 10). Після створення проєкту відкривається файл *Program.cs*, який буде містити код програми.

Для роботи з моделями машинного навчання (ModelBuilder) необхідно додати пакет Microsoft.ML. В меню Visual Studio вибираємо: Project → Manage NuGet Packages… → Browse, вводимо Microsoft.ML і встановлюємо.

Після успішної інсталяції в проєкті з'являться залежності ML.NET, необхідні для роботи з текстовими даними.

Підготовка структури вхідних даних.

Для навчання моделі створюємо два класи для інтерпретації даних:

SceneData – для текстового опису сцени;

ScenePrediction – для збереження результату прогнозу:

```
SceneData.cs*  Program.cs
ConsoleApp5                                   ScenePrediction
1   using Microsoft.ML.Data;
2
    0 references
3   public class SceneData
4   {
5       [LoadColumn(0)]
        0 references
6       public string SceneText { get; set; } = string.Empty;
7
8       [LoadColumn(1)]
        0 references
9       public string Label { get; set; } = string.Empty;
10  }
11
    0 references
12  public class ScenePrediction
13  {
14      [ColumnName("PredictedLabel")]
        0 references
15      public string PredictedLabel { get; set; } = string.Empty;
16  }
17
```

Після створення моделей даних необхідно підготувати навчальний файл *scenes.csv*, який містить текстові описи сцен та відповідні категорії:

```
scenes.csv  SceneData.cs  Program.cs
1   SceneText;Label
2   people with luggage and bags near an airplane;Travels
3   a person at a table with food and tableware;Food
4   a person with a dog near a house;Animals_and_pets
5   a person on a sofa with a cup;Portrait
6   a person at computer in office;Business
7   a person near a painting in a gallery;Art
8   a person on a bicycle on the street;Sport
9   a person near a building with a car;Architecture
10  people with backpacks near a tent in the forest;Travels
11  a group of persons at a table with plates and glasses;Food
12
```

У файлі *Program.cs* (Додаток Б) створюємо об’єкти ML-контексту, завантажуємо дані, виконуємо перетворення тексту (Tokenize → FeaturizeText) та налаштовуємо алгоритм навчання.

Опис має бути англійською мовою та містити 1–2 ключові об’єкти або дії, які характерні для вибраної категорії. Наприклад: *“a person cooking in the kitchen”*, *“dog playing in the yard”*, *“tourists with backpacks near the mountains”*. Для кожного опису потрібно вказати відповідну категорію (Label). Для здобування ознак формується повний набір описів (не менше 100), враховуючі міжкласову схожість і внутрішньокласові розбіжності, щоб модель могла навчитися розпізнавати основні патерни в тексті.

У даному прикладі використовується алгоритм SdcaMaximumEntropy, що добре підходить для багатокласової класифікації тексту.

PipeLine виконує: завантаження текстових даних; перетворення описів у числові вектори ознак; навчання моделі; оцінювання моделі.

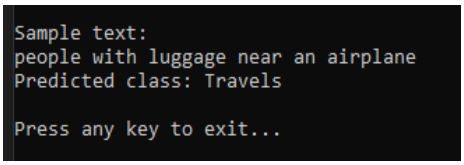
Після запуску програми у консолі відображаються:

MicroAccuracy – середня точність по всіх класах;

MacroAccuracy – середня точність між класами;

LogLoss – показник помилки моделі.

Метрики точності залежать від збалансованості набору даних. Результати навчання моделі у консолі демонструють здатність моделі коректно передбачити категорію сцени:




```
MicroAccuracy = 0.056;
MacroAccuracy = 0.117;
LogLoss = 0.225;
```

Програма також містить демонстрацію роботи моделі під час класифікації нового текстового опису.

У прикладі використовується текст:

“people with luggage near an airplane”

Результат прогнозу: Predicted class: Travels

Це свідчить, що модель змогла розпізнати ключові лінгвістичні ознаки або візуальні відношення (“people with luggage”, “people near an airplane”, “people traveling”) та правильно класифікувати їх із категорією *Travels*. Такі статистично сталі відношення «object – subject – relation» формалізують семантичні залежності, вбудовані в модель машинного навчання на рівні ознак (властивостей) об’єктів і прототипів.

В результаті виконання програми:

– розроблена повноцінна модель класифікації текстових описів сцен;

– отримано показники точності навчання;

– продемонстровано приклад класифікації нової сцени.

Модель коректно визначила категорію, що підтверджує працездатність підходу навіть на невеликій вибірці за рахунок вбудови знань.

Приклад показує, як за допомогою ML.NET можна будувати ефективні моделі для програмних систем аналізу сцен, які використовують текстові описи, сформовані на основі методів прототип-орієнтованого подання знань. Такий підхід дозволяє оперувати з невизначеністю у поданні ознак сцени та скоротити час обробки [14, 15].

Контрольні питання:

1. Які структурні елементи формують опис сцени у прототип-орієнтованому підході?

2. Що таке гранулярні структури та яку роль вони відіграють у реконструкції сцени?

3. Яким чином лінгвістичний опис сцени може бути використаний для машинного навчання?

4. Які методи перетворення тексту використовує ML.NET у задачі багатокласової класифікації?

5. Які критерії визначають точність моделі класифікації сцен?

6. Як впливає різноманітність описів сцени на точність прогнозування?

7. У чому полягають переваги текстового представлення сцени для мобільних систем?

8. Які обмеження має класифікація сцен за текстовим описом у порівнянні з класифікацією за зображенням?

9. Які типи прототипів можуть використовуватись для опису сцен?

10. Як визначити, що сцена містить кілька конкурентних прототипів, і як це впливає на класифікацію?

Додаток А

Таблиця А1 – Набори даних для задач машинного навчання [23]

Dataset	Ознаки / об’єм вибірки	Опис
https://archive.ics.uci.edu/dataset/320/student+performance	30 649	Прогнозування успішності студентів
https://archive.ics.uci.edu/dataset/162/forest+fires	12 517	Прогнозування площі лісових пожеж
https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones	10299	Прогнозування активності за даними смартфону
https://archive.ics.uci.edu/dataset/19/car+evaluation	6 1728	Оцінка вартості авто
https://archive.ics.uci.edu/dataset/360/air+quality	15 9358	Оцінка якості повітря
https://archive.ics.uci.edu/dataset/45/heart+disease	13 303	Прогнозування ризику хвороби серця
https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset	13 17389	Прогнозування кількості оренди велосипедів
https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption	9 2075259	Прогнозування споживання електроенергії
https://archive.ics.uci.edu/dataset/228/sms+spam+collection	5574	Класифікація спаму
https://archive.ics.uci.edu/dataset/27/credit+approval	15 690	Ризик оформлення кредиту
https://archive.ics.uci.edu/dataset/477/real+estate+valuation+data+set	6 414	Оцінка нерухомості
http://archive.ics.uci.edu/dataset/327/phishing+websites	30 11055	Аналіз Phishing websites
http://archive.ics.uci.edu/dataset/332/online+news+popularity	3000	Емоційне забарвлення тексту
http://archive.ics.uci.edu/dataset/691/cifar+10	60000	Класифікація зображень
http://archive.ics.uci.edu/dataset/50/image+segmentation	19 2310	Сегментація зображень
http://archive.ics.uci.edu/dataset/770/nasa+flood+extent+detection	50000	Класифікація зображень сцен
http://archive.ics.uci.edu/dataset/333/forest+type+mapping	30000	Дистанційне зондування

Додаток Б
Вихідний код програми для побудови та тестування моделі аналізу сцен у ML.NET

Вихідний код файлу *Program.cs*

```
using Microsoft.ML;

// Створюємо ML-контекст (вхідна точка для ML.NET)
var mlContext = new MLContext(seed: 1);

// Шлях до CSV з описами сцен
string dataPath = Path.Combine(Environment.CurrentDirectory,
"scenes.csv");

// Завантажуємо дані
IDataView dataView = mlContext.Data.LoadFromTextFile<SceneData>(
    path: dataPath,
    hasHeader: true,
    separatorChar: ';');

// Ділимо на train / test
var split = mlContext.Data.TrainTestSplit(dataView,
            testFraction: 0.2);

// Будуємо конвеєр: Label → key, текст → ознаки, тренер, назад до те-
кстової мітки
var pipeline = mlContext.Transforms.Conversion.MapValueToKey("Label",
"Label")
      .Append(mlContext.Transforms.Text.FeaturizeText(
          outputColumnName: "Features",
          inputColumnName: nameof(SceneData.SceneText)))

.Append(mlContext.MulticlassClassification.Trainers.SdcaMaximumEntropy(
          labelColumnName: "Label",
          featureColumnName: "Features"))

.Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel"));

// Навчаємо модель
Console.WriteLine("Training model...");
var model = pipeline.Fit(split.TrainSet);

// Оцінюємо точність
Console.WriteLine("Evaluating model...");
var predictions = model.Transform(split.TestSet);
var metrics = mlContext.MulticlassClassification.Evaluate(
    predictions,
    labelColumnName: "Label",
    predictedLabelColumnName: "PredictedLabel");

Console.WriteLine($"MicroAccuracy: {metrics.MicroAccuracy:P2}");
Console.WriteLine($"MacroAccuracy: {metrics.MacroAccuracy:P2}");
Console.WriteLine($"LogLoss: {metrics.LogLoss:F4}");
Console.WriteLine();
```

```
    // Створюємо PredictionEngine для одиничних прогнозів
    var predEngine = mlContext.Model.CreatePredictionEngine<SceneData,
ScenePrediction>(model);

    // Приклад: новий опис сцени
    var sample = new SceneData
    {
        SceneText = "people with luggage near an airplane"
    };

    var prediction = predEngine.Predict(sample);

    Console.WriteLine("Sample text:");
    Console.WriteLine(sample.SceneText);
    Console.WriteLine($"Predicted class: {prediction.PredictedLabel}");

    Console.WriteLine();
    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
```

Файл: SceneData.cs

```
using Microsoft.ML.Data;

public class SceneData
{
    [LoadColumn(0)]
    public string SceneText { get; set; } = string.Empty;

    [LoadColumn(1)]
    public string Label { get; set; } = string.Empty;
}

public class ScenePrediction
{
    [ColumnName("PredictedLabel")]
    public string PredictedLabel { get; set; } = string.Empty;
}
```

ПЕРЕЛІК ПОСИЛАНЬ

1. Машинне навчання : навч. посіб. / за наук. ред. д.т.н., проф. В. В. Пасічника, Т. М. Басюк, В. В. Литвин, Л. М. Захарія, Н. Е. Кунанець. 3-тє вид. Львів : Новий Світ-2000, 2026. 330 с.
2. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця : ВНТУ, 2024. 263 с.
3. Штовба С. Д., Козачко О. М. Machine learning: стартовий курс. Вінниця : ВНТУ, 2020. 81 с.
4. Raff E. Inside Deep Learning: Math, Algorithms, Models. Manning Publ., 2022. 600 p.
5. Grohs Ph., Kutyniok G. Mathematical Aspects of Deep Learning. Cambridge University Press, 2023. 492 p.
6. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press book, 2019. 625 p.
7. Литвин В. В., Пелещак Р. М., Висоцька В. А. Глибинне навчання. Львів : Видавництво Львівська політехніка, 2021. 264 с.
8. Литвин В. В. Методи та засоби інженерії даних та знань. Львів : Магнолія, 2021. 242 с.
9. Пасічник В. В. Інтелектуальні системи. Київ : Новий світ, 2021. 406 с.
10. Glassner A. Deep Learning: A Visual Approach. Starch Press, 2021. 776 p.
11. Roberts D. A., Yaida Sh., Hanin B. The Principles of Deep Learning Theory: An Effective Theory Approach to Understanding Neural Networks, Cambridge University Press, 2022. 472 p.
12. Kapoor A., Gulli A., Pal S. Deep Learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models. 3rd ed. Packt Publishing, 2021. 698 p.
13. ML.NET: Machine learning for .NET. URL: https://dotnet.microsoft.com/en-us/apps/ai/ml-dotnet (Last accessed: 18.11.2025)
14. Rakytyanska H. Knowledge Distillation in Granular Fuzzy Models by Solving Fuzzy Relation Equations. In: Pedrycz W., Chen SM. (eds) *Advancements in Knowledge Distillation: Towards New Horizons of Intelligent Systems. Studies in Computational Intelligence*. Springer, Cham, 2023. Vol 1100. P. 95-133. URL: https://doi.org/10.1007/978-3-031-32095-8_4.(Last accessed: 15.11.2025).
15. Rakytyanska H., Prus B. Constructing prototype-based granular fuzzy rules for scene classification on mobile devices. In: S. Babichev, V. Lytvynenko (Eds.). *Lecture notes in data engineering, computational intelligence, and decision-making.* Vol. 1. ISDMCI 2024 ; Lecture Notes on Data Engineering

and Communications Technologies. Cham: Springer, 2024. Vol. 219. P. 194-218. DOI:10.1007/978-3-031-70959-3_10.

16. Kaggle Datasets. URL: https://www.kaggle.com/datasets/ lakshmi25npathi/bike-sharing-dataset (Last accessed: 15.12.2025).

17. MatrixFactorization_MovieRecommendation URL: https://github.com/dotnet/machinelearning-samples/tree/main/samples/csharp/getting-started/MatrixFactorization_MovieRecommendation/Data (Last accessed: 15.12.2025).

18. Kaggle Datasets. URL: https://www.kaggle.com/datasets/abidmeeraj/yelp-labelled-dataset (Last accessed: 15.12.2025).

19. TensorFlow. URL: https://www.tensorflow.org (Last accessed: 15.11.2025).

20.Core ML. URL: https://developer.apple.com/documentation/coreml (Last accessed: 18.12.2025).

21. OpenCV. URL: https://opencv.org/ (Last accessed: 25.12.2025).

22. Kaggle Datasets. URL: https://www.kaggle.com/datasets/imsparsh/flowers-dataset (Last accessed: 15.12.2025).

23. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ (Last accessed: 15.12.2025).

Електронне навчальне видання

Ганна Борисівна Ракитянська

Методичні вказівки до виконання практичних робіт з дисципліни «Розробка проєктів засобами платформи.NET». Частина 1. «Фреймворк машинного навчання ML.NET» зі спеціальності «Інженерія програмного забезпечення»

Рукопис оформила *Г. Б. Ракитянська*

Редактор *Т. Савчук*

Оригінал-макет виготовлено в *РВВ ВНТУ*

Підписано до видання 09.02.2026 р.
Гарнітура Times New Roman.
Зам. № Р2026-016

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
Email: rvv.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.